FIG. 1

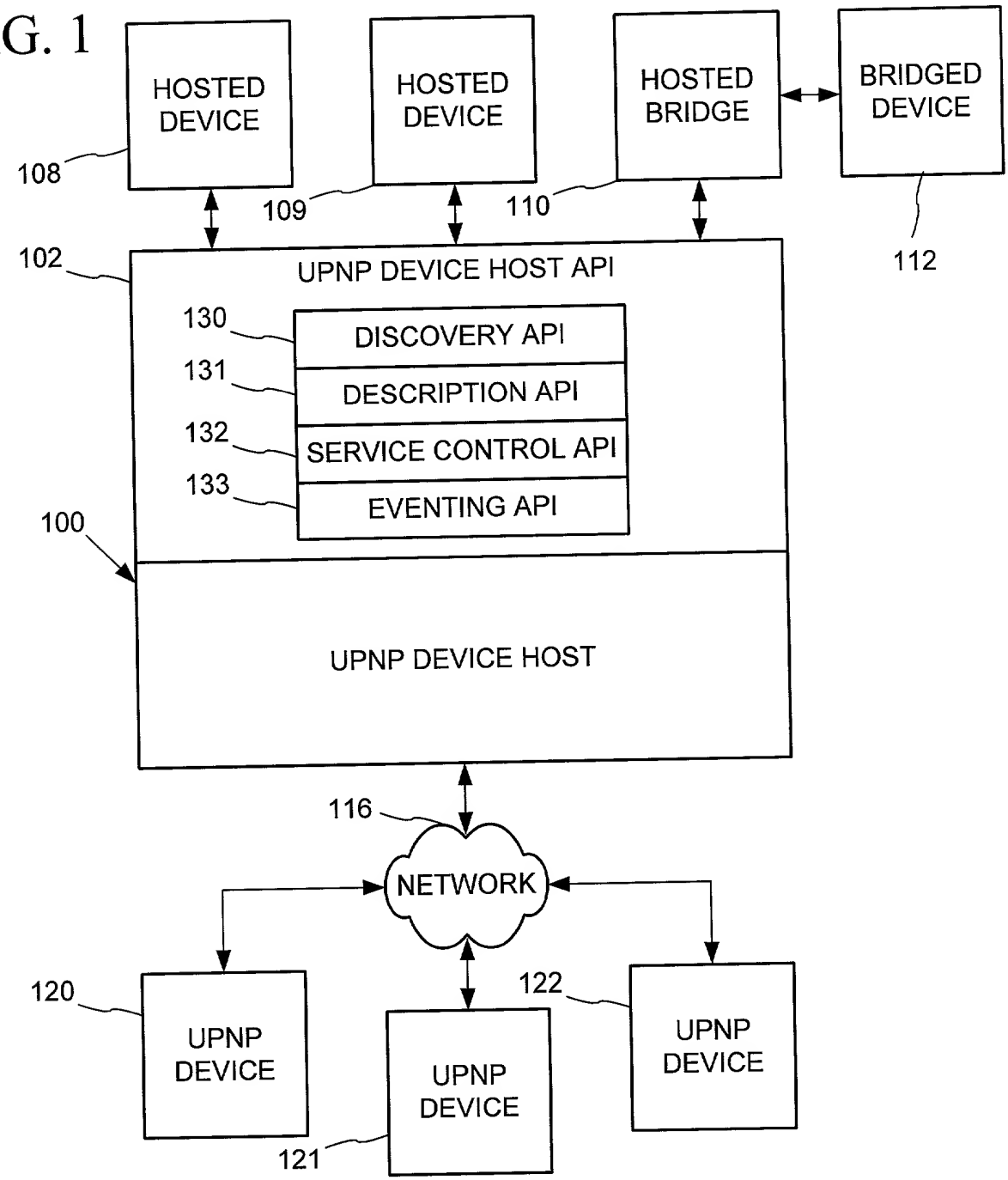


FIG. 2

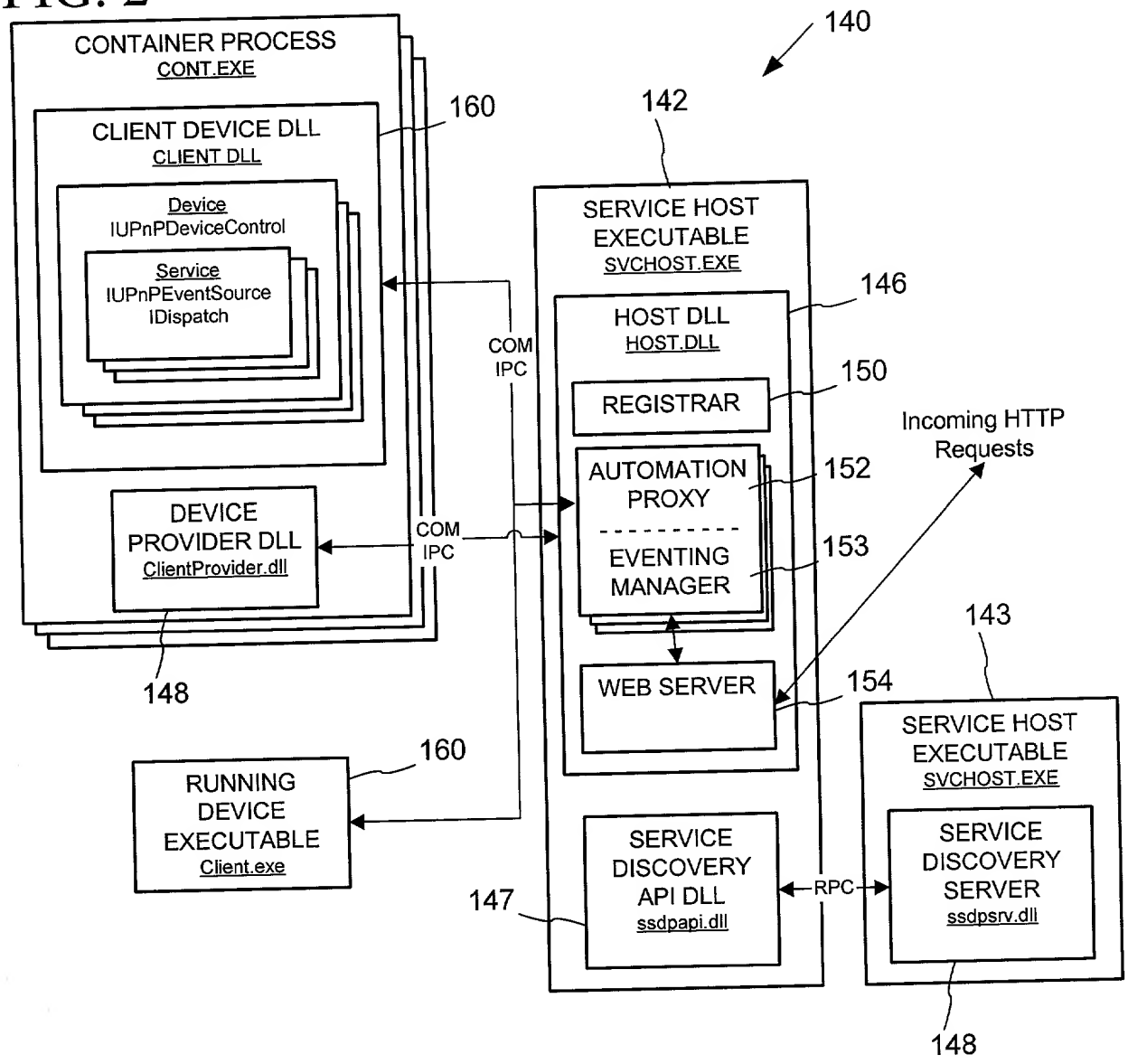

140
CONTAINER PROCESS
CONT.EXE
160
CLIENT DEVICE DLL
CLIENT DLL
Device
IUPnPDeviceControl
Service
IUPnPEventSource
IDispatch
DEVICE
PROVIDER DLL
ClientProvider.dll
148
COM
IPC
COM
IPC
142
SERVICE HOST
EXECUTABLE
SVCHOST.EXE
146
HOST DLL
HOST.DLL
150
REGISTRAR
AUTOMATION
PROXY
152
EVENTING
MANAGER
153
WEB SERVER
154
Incoming HTTP
Requests
143
SERVICE HOST
EXECUTABLE
SVCHOST.EXE
SERVICE
DISCOVERY
SERVER
ssdpsrv.dll
148
RPC
SERVICE
DISCOVERY
API DLL
ssdpapi.dll
147
160
RUNNING
DEVICE
EXECUTABLE
Client.exe

FIG. 3

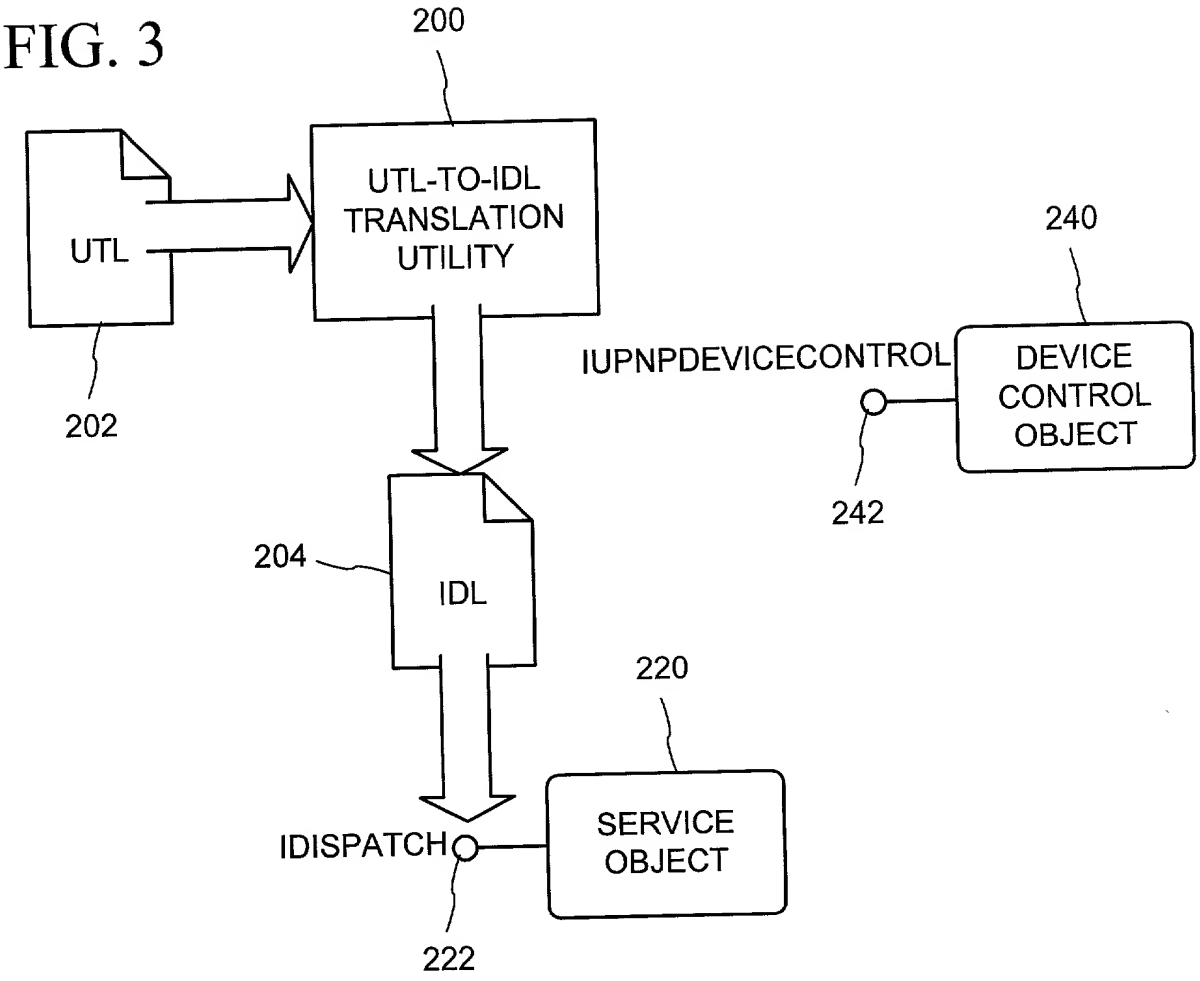

200
UTL
UTL-TO-IDL TRANSLATION UTILITY
202
240
IUPNPDEVICECONTROL
DEVICE CONTROL OBJECT
242
204
IDL
220
IDISPATCH
SERVICE OBJECT
222

FIG. 4

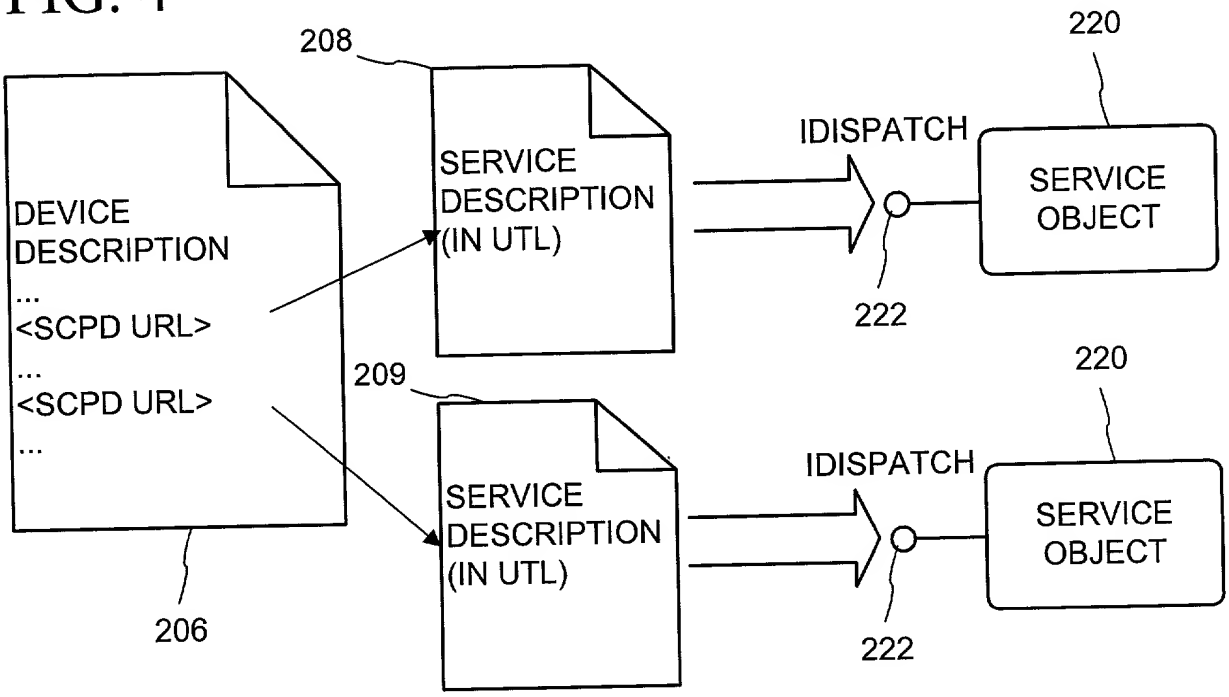

208
DEVICE DESCRIPTION
...
<SCPD URL>
...
<SCPD URL>
...
206
SERVICE DESCRIPTION (IN UTL)
220
IDISPATCH
SERVICE OBJECT
222
209
SERVICE DESCRIPTION (IN UTL)
220
IDISPATCH
SERVICE OBJECT
222

FIG. 5

```
<?xml version="1.0"?>
<scpd xmlns="x-schema:UTL-schema.xml">
 <serviceStateTable>
  <stateVariable>
   <name>Chapter</name>
   <dataType>i4</dataType>
   <allowedValueRange>
    <minimum>1</minimum>
    <maximum>50</maximum>
    <step>1</step>
   </allowedValueRange>
   <defaultValue>0</defaultValue>
  </stateVariable>
 </serviceStateTable>

 <actionList>
  <action>
   <name>GotoChapter</name>
   <argumentList>
    <argument>
     <name>NewChapter</name>
     <direction>in</direction>
     <relatedStateVariable>Chapter</relatedStateVariable>
    </argument>
    <argument>
     <name>OldChapter</name>
     <direction>out</direction>
     <relatedStateVariable>Chapter</relatedStateVariable>
    </argument>
   </argumentList>
  </action>
 </actionList>
</scpd>
```

FIG. 6

```
[
  object
  uuid(ad8b5af4-93fe-4f21-8ff9-a569dc8fcf26),
  helpstring("avtransport-0-1 interface"),
  oleautomation
]
interface IUPnPService_avtransport_0_1
{
  // State Variables (properties)
  [propget] HRESULT Chapter([out, retval] INT * pval);

  // Actions (methods)
  HRESULT GotoChapter ([in, out] INT * pChapter);

}

[
  uuid(8944228e-746b-439b-83ba-089d141eaf6f)
]
library UPNPPrivLib
{
  [
    uuid(7e46fad0-e36b-4be9-b7f2-55c8c075ee4a),
    helpstring("avtransport-0-1 dispinterface"),
    nonextensible
  ]
  dispinterface DIUPnPService_avtransport_0_1
  {
    interface IUPnPService_avtransport_0_1;
  };
};
```

FIG. 7

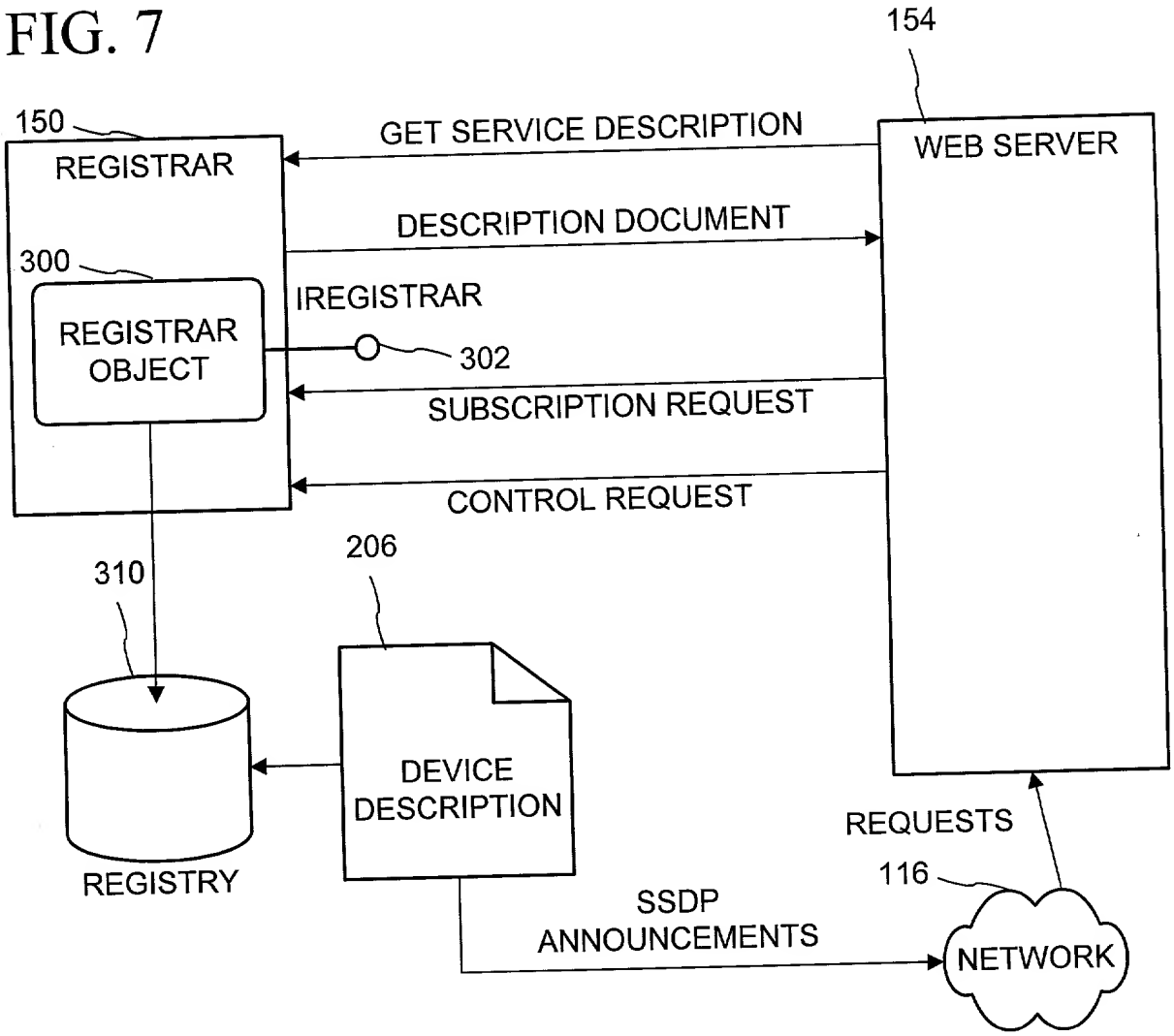

154
150
REGISTRAR
GET SERVICE DESCRIPTION
WEB SERVER
DESCRIPTION DOCUMENT
300
REGISTRAR OBJECT
IREGISTRAR
302
SUBSCRIPTION REQUEST
CONTROL REQUEST
310
206
DEVICE DESCRIPTION
REGISTRY
REQUESTS
116
SSDP ANNOUNCEMENTS
NETWORK

FIG. 8

```
[
  uuid(bebc3352-9eb8-4736-b1eb-093daa00d113),
  helpstring("UPnPRegistrar Class")
]
coclass UPnPRegistrar
{
  [default] interface IUPnPRegistrar;
  interface IUPnPReregistrar;
};
```

FIG. 9

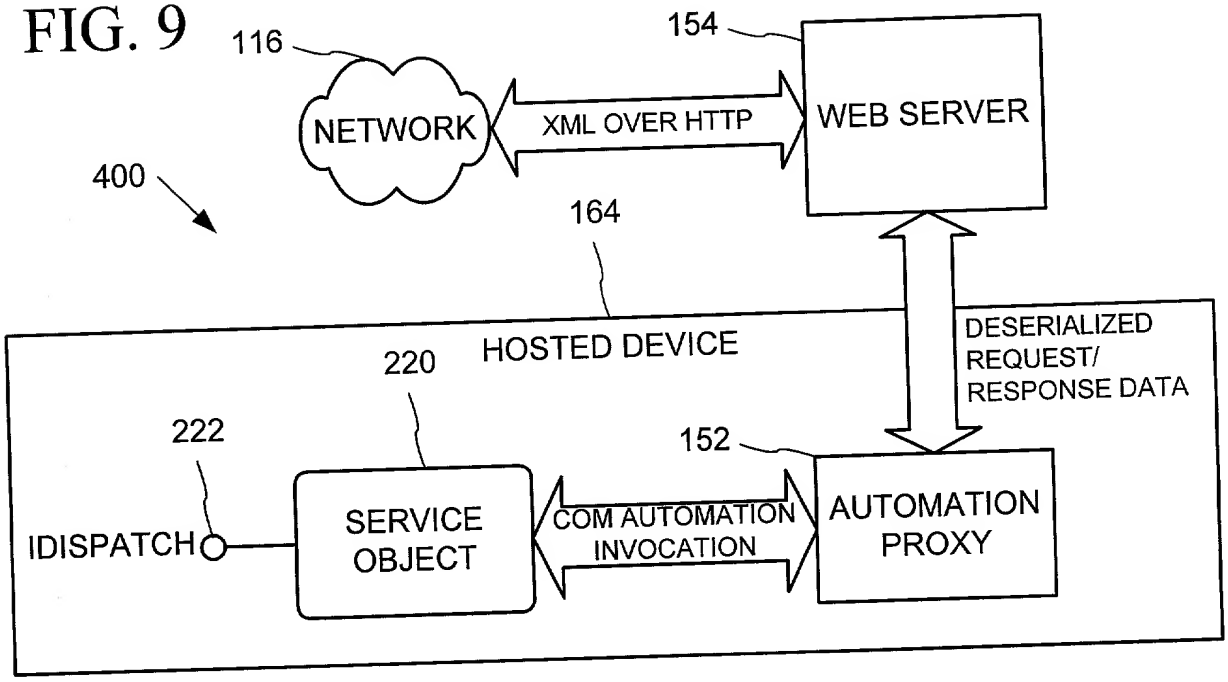

116
154
NETWORK
XML OVER HTTP
WEB SERVER
400
164
HOSTED DEVICE
DESERIALIZED
REQUEST/
RESPONSE DATA
220
222
152
IDISPATCH
SERVICE
OBJECT
COM AUTOMATION
INVOCATION
AUTOMATION
PROXY

FIG. 10

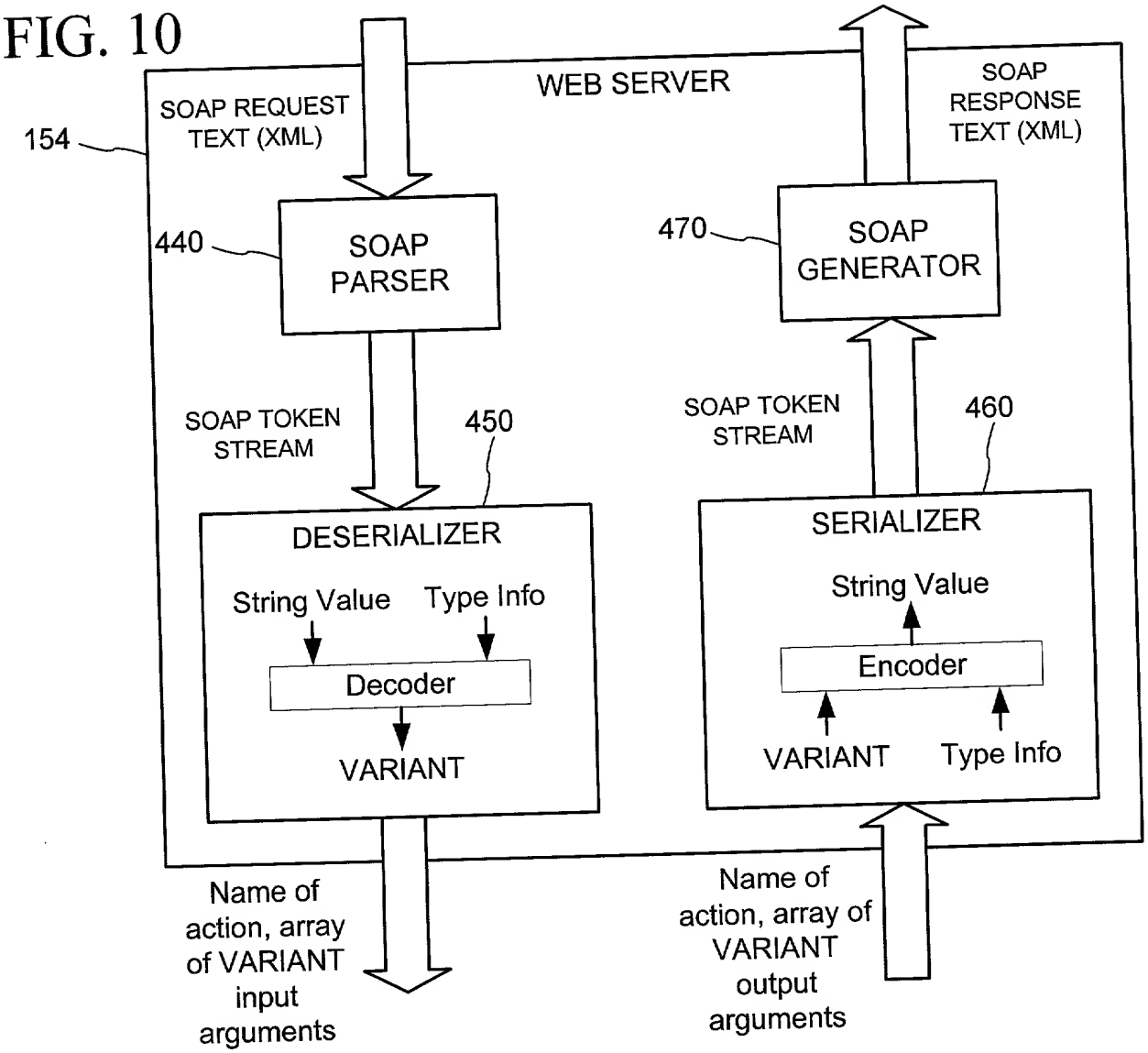


FIG. 11

```
struct tagUPNP_SOAP_REQUEST
{
BSTR                bstrActionName
IXMLDOMNodeList * pxdnlArgs
} UPNP_SOAP_REQUEST;
```

FIG. 12

```
struct tagUPNP_CONTROL_REQUEST
{
BSTR        bstrActionName;
DWORD               cInputArgs;
VARIANT     rgvarInputArgs[];
} UPNP_CONTROL_REQUEST;
```

FIG. 13

```
struct tagUPNP_CONTROL_RESPONSE
{
        BSTR bstrActionName;
union
{
struct
{
DWORD               cOutputArgs;
VARIANT     rgvarOutputArgs[];
        };
struct
{
BSTR        bstrFaultCode;
BSTR        bstrFaultString;
DWORD               dwUPnPErrorCode;
BSTR        bstrUPnPErrorString;
        };
};
} UPNP_CONTROL_RESPONSE;
```

FIG. 14

```
Deserialize(
   UPNP_SOAP_REQUEST            * pusr,
   IUPnPServiceDescriptionInfo   * pusdi,
   UPNP_CONTROL_REQUEST          * pucr)
{
IXMLDOMNode * pxdnArgument = NULL;

// Set up UPNP_CONTROL_REQUEST structure.

pucr->bstrActionName = SysAllocString(pusr->bstrActionName);
pucr->cInputArgs = pusr->pxdnlArgs->length;
pucr->rgvarInputArgs = AllocateMemory(pucr->cInputArgs *
                              sizeof(VARIANT));

// Deserialize each argument

DWORD i = 0;

for each pxdnArgument in pusr->pxdnlArgs
{
VARIANT varDataType;

VariantInit(&varDataType);
varDataType.vt = VT_BSTR;
V_BSTR(&varDataType) =
   pusdi->GetArgumentType(pucr->bstrActionName,
                     pxdnArgument.name);

pxdnArgument->put_dataType(varDataType);
pxdnArgument->get_nodeTypedValue(&pucr->rgvarInputArgs[i]);
i++;
}
}
```

FIG. 15

```
struct tagUPNP_SOAP_RESPONSE
{
union
{
struct
{
BSTR        bstrActionName;
IXMLDOMNodeList * pxdnlOutArgs;
};
struct
{
BSTR        bstrFaultCode;
BSTR        bstrFaultString;
BSTR        bstrUPnPErrorCode;
BSTR        bstrUPnPErrorString;
};
}
} UPNP_SOAP_RESPONSE;
```

FIG. 16

```
[
        uuid(0FB40F0D-1021-4022-8DA0-AAB0588DFC8B),
        helpstring("AutomationProxy Class")
]
coclass UPnPAutomationProxy
{
        [default] interface IUPnPAutomationProxy;
        interface IUPnPServiceDescriptionInfo;
};
```

FIG. 17

```
[
   object,
   uuid(de2eca21-99d0-4d92-b2ae-cb994e85f31f),
   pointer_default(unique),
   version(1.0)
]

interface IUPnPEventingManager : IUnknown
{
  [helpstring("method Initialize")]
  HRESULT Initialize(
          [in] LPCWSTR                    szUdn,
          [in] LPCWSTR                    szSid,
          [in] IUPnPAutomationProxy *  puap,
     [in] IUnknown *          punkSvc);
  [helpstring("method AddSubscriber")]
  HRESULT AddSubscriber(
     [in] LPWSTR    szCallbackUrl,
     [in] DWORD *   pcsecTimeout,
          [out] LPWSTR *  pszSid);
  [helpstring("method RenewSubscriber")]
  HRESULT RenewSubscriber(
     [in] DWORD     *       pcsecTimeout,
     [in] LPWSTR    szSid);
  [helpstring("method RemoveSubscriber")]
  HRESULT RemoveSubscriber(
     [in] LPWSTR  szSid);
  [helpstring("method Shutdown")]
  HRESULT Shutdown();
}

interface IUPnPEventSink: IUnknown
{
  [helpstring("method OnStateChanged"), hidden]
  HRESULT OnStateChanged(
              [in] DWORD cChanges,
              [in, size_is(cChanges)] DISPID rgdispidChanges[]);
  [helpstring("method OnStateChangedSafe")]
  HRESULT OnStateChangedSafe(
              [in] SAFEARRAY(DISPID) rgdispidChanges);
}
```

FIG. 18

```
[
    object,
    uuid(b05e0973-c342-4650-9d04-9224c6e6ded9),
    pointer_default(unique),
    version(1.0)
]

interface IUPnPEventSource: IUnknown
{
  [helpstring("method Advise")]
  HRESULT Advise(
            [in] IUnknown *  punkSubscriber);

  [helpstring("method Unadvise")]
  HRESULT Unadvise(
     [in] IUnknown * punkSubscriber);
}
```

FIG. 19

```
struct UPNP_SUBSCRIBER
{
        UPNP_EVENT_SOURCE *     pes;            // Pointer to event source
        LPWSTR                  szUrl;          // Callback URL
        FILETIME                ftTimeout;      // Timeout period
        DWORD                   csecTimeout;    // Timeout period
        DWORD                   iSeq;           // Event sequence number
        LPTSTR                  szSid;          // Subscription Identifier
        DWORD                   cRenewals;      // # of renewals received
        HANDLE                  hEventQ;        // Event signaled when Q full
        UPNP_EVENT *            pueQueue;       // Event queue
        HANDLE                  hWait;          // Handle of registered wait
        UPNP_SUBSCRIBER *       psubNext;       // Next subscriber in list
};

struct UPNP_EVENT_SOURCE
{
        LPWSTR                  szEsid;         // Event source identifier
        UPNP_SUBSCRIBER *       pusList;        // List of subscribers
        HANDLE                  hTimerQ;        // Handle to timer queue
        UPNP_EVENT_SOURCE *     pesNext;        // Next in list
        DWORD                   csecTimeout;    // Preferred timeout
};
struct UPNP_RENEWAL
{
        LPWSTR                  szEsid;         // Event source identifier
        LPTSTR                  szSid;          // Subscription Identifier
        DWORD                   iRenewal;       // renewal index
};
struct UPNP_WAIT_PARAMS
{
        LPWSTR                  szEsid;         // Event source identifier
        LPTSTR                  szSid;          // Subscription Identifier
};

struct UPNP_EVENT
{
        LPWSTR                  szBody;         // body of event message
};
```

FIG. 20

```
HRESULT HrRegisterEventSource(LPCWSTR szEsid);

HRESULT HrDeregisterEventSource(LPCWSTR szEsid);

HRESULT HrAddSubscriber(
        LPCWSTR             szEsid,
        LPCWSTR             szCallbackUrl,
        LPCWSTR             szEventBody,
        DWORD *     pcsecTimeout,
        LPWSTR      *       pszSid);

HRESULT HrRenewSubscriber(
        LPCWSTR             szEsid,
        DWORD *     pcsecTimeout,
        LPCWSTR             szSid);

HRESULT HrRemoveSubscriber(
        LPCWSTR             szEsid,
        LPCWSTR             szSid);

HRESULT HrSubmitEvent(LPCWSTR szEsid,
                      LPCWSTR szEventBody);

HRESULT HrSubmitEventZero(LPCWSTR szEsid, LPCWSTR szSid,
                          LPCWSTR    szEventBody);
```

FIG. 21

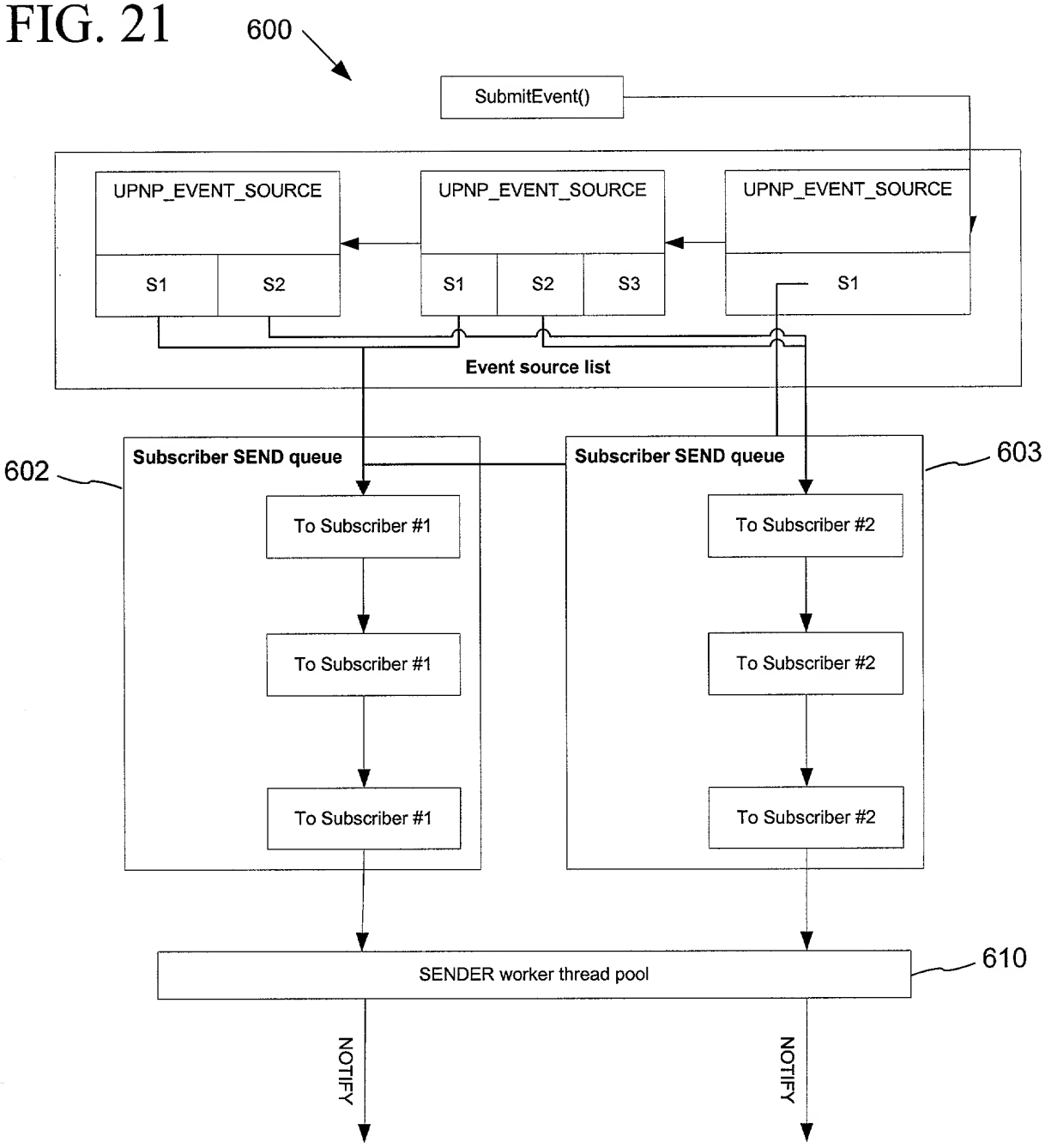

600
SubmitEvent()
UPNP_EVENT_SOURCE
S1
S2
UPNP_EVENT_SOURCE
S1
S2
S3
UPNP_EVENT_SOURCE
S1
Event source list
602
Subscriber SEND queue
603
Subscriber SEND queue
To Subscriber #1
To Subscriber #1
To Subscriber #1
To Subscriber #2
To Subscriber #2
To Subscriber #2
610
SENDER worker thread pool
NOTIFY
NOTIFY

FIG. 22

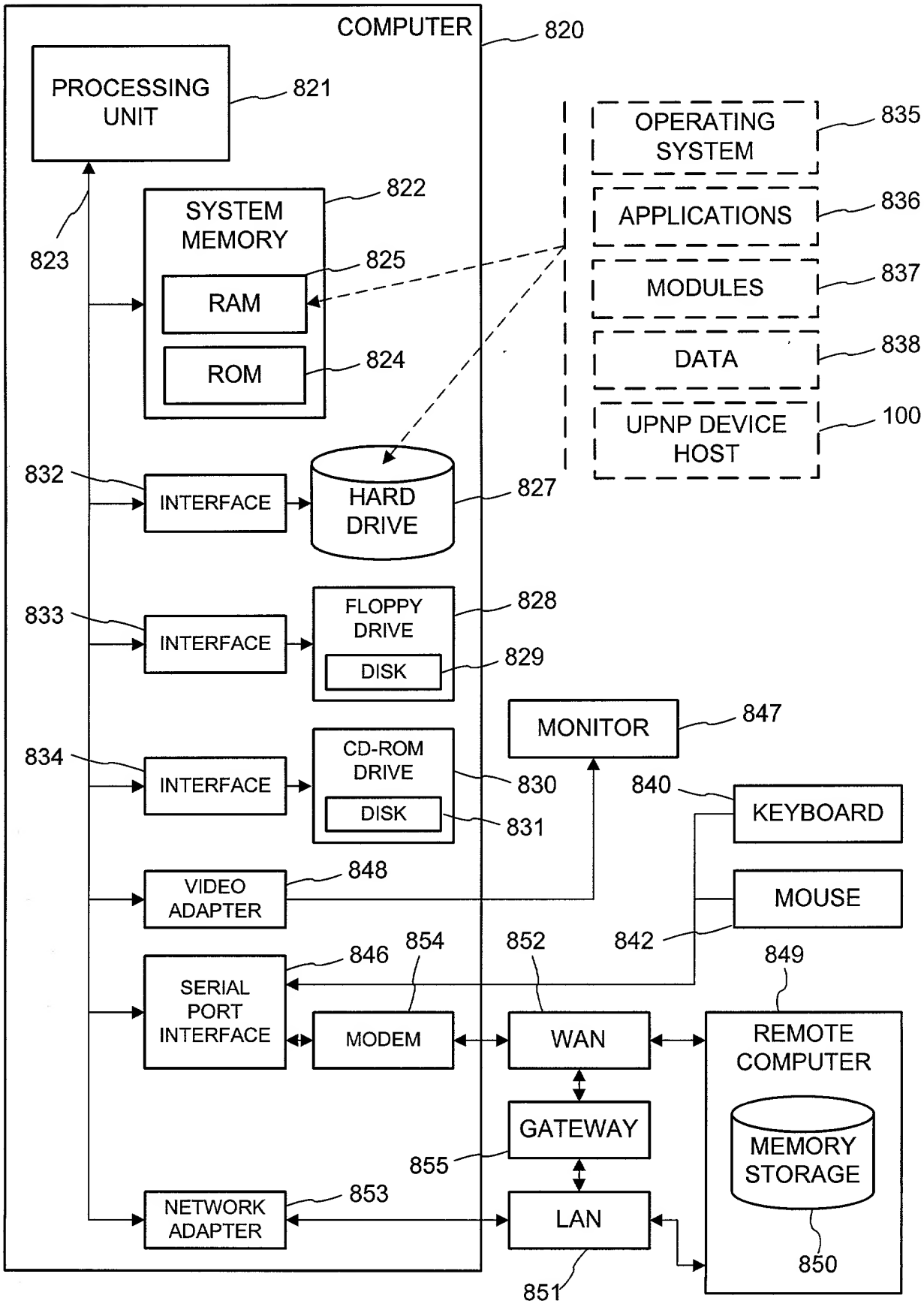

COMPUTER
820
PROCESSING UNIT
821
SYSTEM MEMORY
822
823
RAM
825
ROM
824
OPERATING SYSTEM
835
APPLICATIONS
836
MODULES
837
DATA
838
UPNP DEVICE HOST
100
832
INTERFACE
HARD DRIVE
827
833
INTERFACE
FLOPPY DRIVE
828
DISK
829
834
INTERFACE
CD-ROM DRIVE
830
DISK
831
MONITOR
847
840
KEYBOARD
MOUSE
842
VIDEO ADAPTER
848
SERIAL PORT INTERFACE
846
854
852
849
MODEM
WAN
REMOTE COMPUTER
GATEWAY
855
MEMORY STORAGE
850
NETWORK ADAPTER
853
LAN
851

FIG. 23

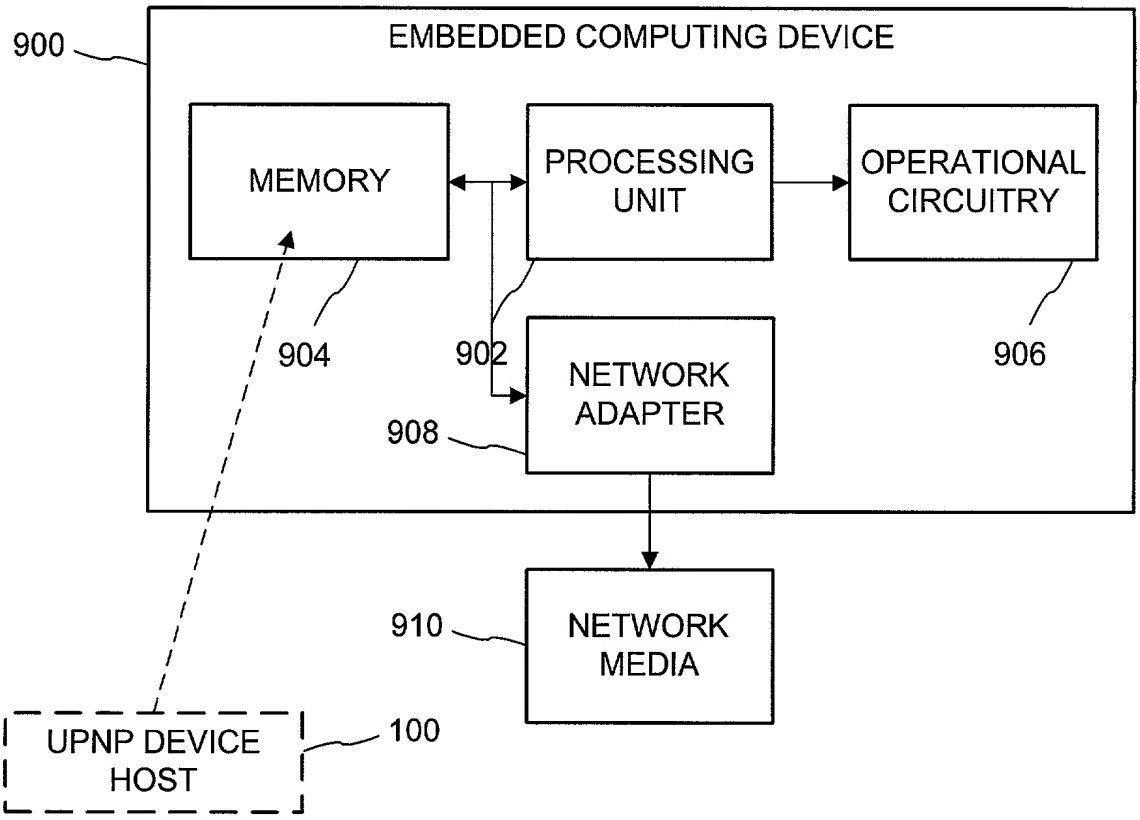
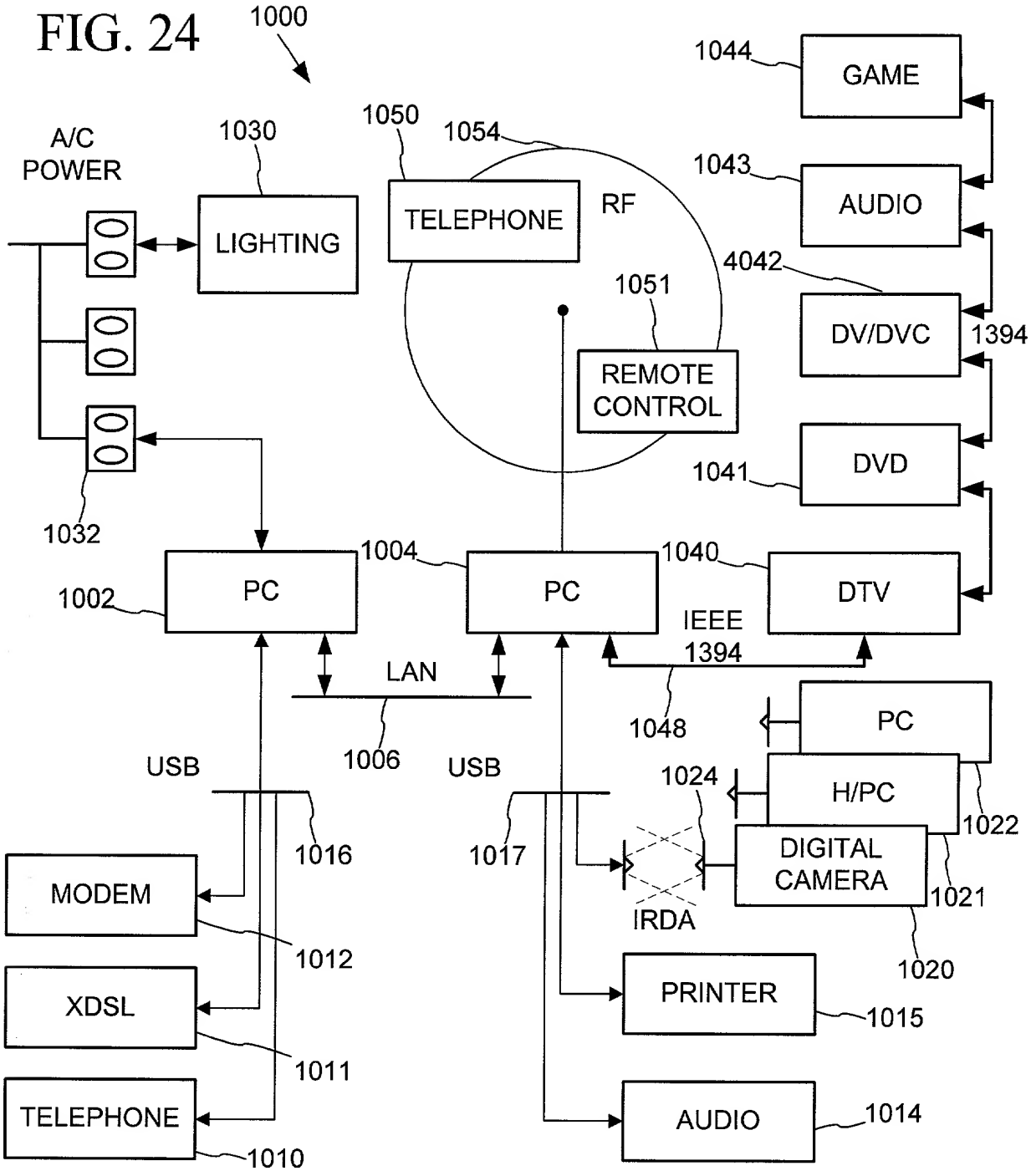

900
EMBEDDED COMPUTING DEVICE
MEMORY
PROCESSING UNIT
OPERATIONAL CIRCUITRY
904
902
906
NETWORK ADAPTER
908
910
NETWORK MEDIA
UPNP DEVICE HOST
100

FIG. 24


1000
1044
GAME
A/C POWER
1030
1050
1054
1043
AUDIO
TELEPHONE
RF
LIGHTING
4042
1051
DV/DVC
1394
REMOTE CONTROL
1041
DVD
1032
1004
1040
1002
PC
PC
DTV
IEEE 1394
LAN
1048
PC
1006
USB
USB
1024
H/PC
1022
1016
1017
DIGITAL CAMERA
1021
MODEM
IRDA
1012
1020
XDSL
PRINTER
1015
1011
TELEPHONE
AUDIO
1014
1010